Introduction to Programming in R

This course is aimed at those who would like to take advantage of the powerful statistical computing software 'R' for their research and data analysis but may have no previous experience of computer programming. The goals of the course are to introduce some general programming concepts, gain an understanding of what R can be used for and hands-on practical experience while promoting programming 'best practice'.

We recommend spending a few hours familiarising yourself with the online tutorial we will follow, in advance of attending the course; the course itself will include discussion of the underlying ideas and time to continue working through the tutorial at your own pace through a mixture of lectures and workshop sessions. We are running it multiple times to accommodate demand and there will be a maximum of around 15 participants to allow for one-to-one interactions and help from the tutors (Helen Lockstone and Ben Wright, Bioinformatics Core at Wellcome Centre for Human Genetics, WHG). We hope you will find it useful to start learning R by explaining the technical aspects in an accessible way, guiding you through its wide-ranging functionality, and providing practical information and knowledge from our many years of experience using R.

You will need to install the R software https://www.r-project.org/ and RStudio (an interface to R with many helpful features) https://www.rstudio.com/products/RStudio/

When you have a confirmed date for the course you will be attending, please bring your own laptop with the installed software and ideally have an active Eduroam account (http://help.it.ox.ac.uk/network/wireless/services/eduroam/index) for internet access (visitor accounts will be available if needed). The location will be WHG on the Old Road Campus.

Getting Started with the 'Software Carpentry' Online Tutorial for R

We are using an open source computing skills workshop provided by the Software Carpentry Foundation (https://software-carpentry.org/about/) as part of this course.

Start by going to the home page of the 'Introduction to Programming in R' Software Carpentry lesson: http://swcarpentry.github.io/r-novice-inflammation/

This gives an overview of the course and a schedule composed of 15 tutorials. Our version will cover 11 selected topics in the order given below (note this is different to that presented on the webpage); we first cover some fundamentals of R programming and later move onto more advanced topics as well as considering best practice when writing code.

Before starting to work through these, click the 'Setup' link for instructions on downloading the files that will be needed for the tutorial.

R Fundamentals
Introduction to RStudio
Analysing Patient Data
Addressing Data
Reading and Writing CSV Files
Data Types and Structures
Understanding Factors

Advanced Topics
Analysing Multiple Datasets
Loops in R
Creating Functions
Best Practices for Writing R
Making Choices

There is a brief orientation to the RStudio interface in the first lesson. The 'Analysing Patient Data' tutorial introduces some fundamental concepts and functions of R while performing the common task of working with some data to understand some of its characteristics. It's helpful to start with this one to get a sense of using R straight away, but at this point don't worry about understanding the detail of each command – the general idea of 'these lines of code produce plot X' and pasting the commands into R to generate it yourself is fine. The subsequent lessons cover specific topics in more detail (e.g. reading and writing files, data types and structures in R, loops and functions) and these you can work through at your own pace (there will also be time during the course to continue with this). Each topic will also be covered via lecture slides and interactive demos on the face-to-face course.

We encourage direct typing of the commands into your R session, rather than copying and pasting them from the web tutorial, because it will help you become familiar with how the commands are constructed (syntax), and which bits you need to modify to change the behaviour. To save re-typing things many times though, you will find the up/down arrow keys useful as they scroll back and forth through previously entered commands - to change one part of a previous command, press the 'up' arrow key and edit it as needed.

When you are ready to start the 'Analysing Patient Data' tutorial, you may find a problem getting the necessary example files loaded into R (this seems to be a Windows-related issue relating to directory paths). Please refer to the detailed section below if you get an error message at this step to see how to resolve it. It's also a perfect example of the precisely correct instructions that need to be provided when programming and how you might go about troubleshooting an error message or unexpected output – R tells you something is not working but you have to work out the cause(s) of the problem, usually using R itself to do some testing and checks. Once they are correctly loaded, the rest of the tutorial should work quite smoothly – if not, see the 'Helpful Hints' section to try and troubleshoot the problem. If you are really stuck and want to make further progress through the tutorial before attending the course, you can email us (hel23@well.ox.ac.uk, bjw78@well.ox.ac.uk).

Just to re-iterate, the idea is to gain a little bit of familiarity with R in advance and it is not necessary to complete or understand all of the sections before attending. Approximately half of the 2-day course will be spent continuing to work through the online tutorial from whichever point you are at, and to give individual help.

Helpful Hints

Over time, you'll identify typical reasons for error messages and have some idea what the nature of the problem is likely to be. If there's not an obvious cause, it can take a bit of work to pin down the exact issue and decide on a good solution. Below are some tips about R's behaviour and possible reasons you might receive error messages.

R Functions

There are hundreds, probably thousands, of R functions and we will become familiar with some of them during this course. There are also numerous add-on packages developed to provide even further functionality. In particular, the BioConductor project https://www.bioconductor.org/) is a collection of packages for handling and analysing genomic data in R and is a key reason for the widespread use of R in this field.

An R function is a piece of pre-written code to perform a common task efficiently, for example to read in a file or calculate a mean value. All in-built R functions are invoked by their name and need to be typed correctly to be recognised as valid function names. Pay attention to where capital letters might be used and the '.' in read.csv, read.table for example. The behaviour of a function is controlled by arguments that are passed to it e.g. the name of the file to read in, whether it has a header row etc. Arguments are given within a pair of brackets following the function name with multiple arguments separated by commas. Some arguments are optional or have a default value that will be used in the absence of specifying anything else. An error may result if necessary arguments are missing or incorrectly specified – to see how they should be used for a particular function, you can look at its help page e.g. for the function 'setwd' you can type help(setwd) or ?setwd() or via the 'Help' menu in RStudio.

Accuracy and consistency

When creating new objects, the programmer decides on a suitable name (which can be almost anything subject to certain rules). Unlike other programming languages, these can been created 'on-the-fly' and don't need to be defined before referring to them.

The following command will create a vector of the numbers 1 to 10 and store them in an object called 'mydata':

```
mydata <- c(1:10)
```

<- is the assignment operator, and c(1:10) means 'concatenate the numbers between 1 and 10. An = sign can be used interchangeably with <- in most situations and is now widely used as the assignment operator. If we slightly change the name, another object called myData is created, containing the same vector of numbers 1 to 10:

```
myData <- c(1:10)
```

mydata and myData are identified as two separate variables by R due to the capital D in the latter. If we assign something else to the first object, the contents will be overwritten (there will be no warning message, R will simply obey the instruction).

```
mydata <- c(1:20) # now 'mydata' holds the numbers 1 to 20
```

If we try to refer to an object that doesn't exist (without assigning anything to it), an error message will result:

```
MyData
Error: object 'MyData' not found
```

If you think it should exist, double-check for typing errors. You can also list all objects in the current session using the following command:

```
objects()
```

Syntax and Error Messages

If a command is not valid in the way it is constructed (its syntax), R will print an error message to the screen. These can sometimes be hard to interpret but particularly common culprits are quotes and brackets, whether they are in the wrong place, missing, or not in pairs (e.g. missing a closing bracket). Take care with brackets (especially when nesting multiple commands) as if they are incorrectly placed, the line of code may be executed but not behave as intended.

The usual R prompt is a > symbol at the start of the line and means R is ready for an input command. Usually after typing your command and pressing enter, R will execute that command and return to the > prompt. If you see a + instead, it means the command is unfinished (as opposed to invalid) – you may need to enter another bracket for example. If you can't fix it easily, you can get back to the usual > by pressing the Esc key to cancel that attempt and start again.

History of commands – you can use the up/down arrow keys to scroll through previous commands to re-run or edit them. The tab key or RStudio features can help auto-fill names of functions, object/variable names and filenames. RStudio also automatically pairs brackets and quotes to help avoid those errors.

The # is the comment character – lines starting with a # (conventionally ##) can include comments about what the code is doing, while entering a # after a command can be useful if you want to note down some information relating to that command – everything after the # will be ignored by R but serve as useful information to the programmer. You can also use long lines of # symbols to break your code into sections.

Working Directory Issue On Windows

If you are using a Windows machine and followed the instructions on the 'Setup' page, you will likely run into a problem when trying to load the files into R using the first two commands given in the 'Analysing Patient Data' lesson:

```
setwd("~/Desktop/r-novice-inflammation/")

read.csv(file = "data/inflammation-01.csv", header = FALSE)
```

Let's look at what they are trying to do. The first command uses the 'setwd' function to define a path to a working directory (the ~ is short-hand for the user's home directory; R will automatically expand it). The second command is trying to read in a file named 'inflammation-01.csv' and the 'data/' in front of the file name is a relative path; it indicates that the file of interest is located in a subfolder named 'data' within the current working directory (i.e. we should set the working directory as the one immediately above the 'data' folder). Setting the relevant working directory for the dataset or project you are working on is usually the first thing you will do in a new R session, along with loading any additional libraries required for your analysis. Both the relevant working directory and the location of files need to be correctly specified with these initial input commands, or an error message will result. There are actually two separate reasons for the error messages that I encountered trying to run the commands above – see the Long Version for a detailed explanation of the problem. It also serves as a good example of how you might troubleshoot an issue you encounter in R.

Short Version

The quickest solution is to ignore the first command of the tutorial and set the working directory using the point and click menu option. In RStudio, click on 'Session' menu, choose 'Set working directory' and 'Choose directory'. You should then be able to navigate to locate the downloaded files. For example, the full path to the files on my machine was:

C:/Users/hel23/Desktop/r-novice-inflammation/r-novice-inflammation-data/data

Note that an extra folder named 'r-novice-inflammation-data' was automatically created when extracting the downloaded zip file. If I set that as the working directory, the second command should then work (upon running it the contents of the file `inflammation-01.csv` will be printed to the screen).

Long Version

The problems relate to determining and setting the relevant working directory. If we enter the first command, R gives us an error message:

```
setwd("~/Desktop/r-novice-inflammation/")
Error in setwd("~/Desktop/r-novice-inflammation/") :
  cannot change working directory
```

The error implies either that directory does not exist or the filepath is wrong in some way (that could be a typo in the names of one of the folders or it is inconsistent with the actual file structure on your computer). We should first double-check the folder we created on the Desktop is named exactly as given in the command. If it is, look at the ~ used in the command. R expands this automatically to your home directory, but what is this defined as?

We can run a reduced version of the setwd command to find out:

```
setwd("~")
```

R should return to prompt with no error message –internally R has changed the working directory to 'home' but we still don't know exactly where that is. A related command will tell us the new working directory:

```
getwd()
```

On my machine, this is C:/Users/hel23/Documents and so the original setwd command was trying to set C:/Users/hel23/Documents/Desktop/r-novice-inflammation/ as my working directory. This doesn't match my directory structure (it is looking for the Desktop folder within the Documents folder while it is actually C:/Users/hel23/Desktop/r-novice-inflammation/).

I can now use setwd with the correct full path to make this folder the working directory or go through the RStudio menu as described above.

```
setwd("C:/Users/hel23/Desktop/r-novice-inflammation/")
```

The next step is a command for reading in the data from one of the files, and the tutorial gives:

```
read.csv(file = "data/inflammation-01.csv", header=FALSE)
```

This assumes that the data files to read in are located in a subdirectory of the current working directory named 'data' – in other words, when we downloaded and unzipped the files for this tutorial, the directory structure produced was C:/Users/hel23/Desktop/r-novice-inflammation/data

Upon entering the read.csv command, I got another error:

```
Error in file(file, "rt") : cannot open the connection
In addition: Warning message:
In file(file, "rt") :
  cannot open file 'inflammation-01.csv': No such file or directory
```

It's quite a long message but the key is in the last line – this file does not exist (at least not where R is looking). When I double-checked, the actual directory structure created by double-clicking on the zipped folder was C:/Users/hel23/Desktop/r-novice-inflammation/r-novice-inflammation-data/data

Quite silently, Windows created an extra subfolder with the name of the zipped folder (r-novice-inflammation-data) and the 'data' folder is in there. We can either change our working directory again to the one named 'r-novice-inflammation-data' for the read.csv command to work, or re-structure our directories to match the tutorial. I chose the latter, as even with filename expansion (pressing the tab key in R will auto-complete file/directory/object names – this is very useful to save typing long names in full), it seemed an unnecessary subfolder to have. So I cut and pasted the 'data' directory one level up, and then removed the empty 'r-novice-inflammation-data' folder.

Finally we have everything correct – our working directory is C:/Users/hel23/Desktop/r-novice-inflammation and within that is a subfolder named 'data' containing files of interest and where we can direct R to read one of them in with:

```
read.csv(file = "data/inflammation-01.csv", header=FALSE)
```

This should now work and print the contents of that file to your R console screen.